strcpy() and strcat()

Daniel Plakosh, Software Engineering Institute [vita[1]]

2005-09-27; Updated 2008-07-17

L4 / D/P, L[2]

The `strcpy()` and `strcat()` functions have been villainized as a major source of buffer overflows, and there are many mitigation strategies that provide more secure variants of these functions. However, not all applications of `strcpy()` are flawed.

Development Context

Copying and concatenating character strings

Technology Context

C, UNIX, Win32

Attacks

Attacker executes arbitrary code on machine with permissions of compromised process or changes the behavior of the program.

Risk

The strcpy() and strcat() functions are a source of buffer overflow vulnerabilities.

Description

The `strcpy()` and `strcat()` routines have been villainized as a major source of buffer overflows, and many prevention strategies provide more secure variants of these functions. However, not all applications of `strcpy()` are flawed. For example, it is often possible to dynamically allocate the required space as follows:

```
dest = (char *)malloc(strlen(source) + 1);
if (dest) {
  strcpy(dest, source);
} else {
  /* handle error */
  ...
}
```

For this example to work, it is necessary that the source string be fully validated; for example, to ensure that the string is not overly long. There are also other cases where it is clear that there is no potential for writing beyond the array bounds.

As a result, it may not be cost effective to replace or otherwise secure every call to `strcpy()`. This depends on the overall mitigation strategy adopted, however, as some strategies require an overall retooling of string manipulation.

1. http://buildsecurityin.us-cert.gov/bsi/about_us/authors/268-BSI.html (Plakosh, Daniel)

References

[ISO/IEC 99] ISO/IEC. *ISO/IEC 9899 Second edition 1999-12-01 Programming languages — C*. International Organization for Standardization, 1999.

Pearson Education, Inc. Copyright